REMARKS/ARGUMENTS

This Amendment is in response to the Office Action mailed May 28, 2008. Claims 1-3 and 6-14 were pending in this application. This Amendment amends claims 1, 8, 10, and 14, and cancels claims 2, 3, 6, and 7. Claims 15 and 16 are new. After entry of this amendment, claims 1 and 8-16 are pending. Reconsideration of the rejected claims and consideration of the newly presented claims is respectfully requested.

I. Rejection under §101

Claim 14 was rejected under 35 U.S.C. §101 because the claimed invention is directed to non-statutory subject matter. Specifically, it is asserted that claim 14 is directed to software per se which is not a process, machine or manufacture, or a composition of matter. (Office Action, p. 2). Without conceding the merits of the rejection as applied to the original claims, Applicants respectfully submit that the amended claim 14 overcomes the rejection. Applicants therefore respectfully request that the rejection with respect to claim 14 be withdrawn.

II. Rejection Under 35 U.S.C. §112, Second Paragraph

Claims 1-3, 6-8, 10 and 14 are rejected under 35 U.S.C. §112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Regarding claim 1, the Examiner states that "the metadata in the database" references other items in the claim. Claim 1 recites "a database…for storing metadata objects," a configuration management module to create "a deployable collection of metadata objects from the metadata objects stored in the database," and a validation engine to "validate the metadata objects stored in the database." Therefore, the Applicants respectfully request the rejection to be withdrawn.

Regarding claims 2, 3, 6, and 7, the Examiner states that the validation subject references at least two other validation subjects. Claims 2, 3, 6 and 7 have been canceled herein. Therefore, the Applicants respectfully request the rejection to be withdrawn.

Regarding claim 8, the Examiner states in claim 8, completeness validation rules are applied to the object instance if a user selects validation, and completeness validation rules are also applied to the object instance automatically. The Examiner states that it is unclear how the one or more completeness validation rules is applied to the object instance. (Office Action, p. 3). Applicants refer the Examiner to paragraph [0009] of the application for example, which states that

> In one aspect, at design time, only correctness type validation is performed. Thus, the present invention advantageously allows for incomplete objects to be created at design time. The developer, however, in this case may opt to perform completeness validation at any time. In general, a developer may opt to perform completeness and/or correctness validation at any time independent of deployment processing. In another aspect, full validation (*e.g.*, completeness and correctness) is automatically performed on the objects prior to deployment processing. (emphasis added).

Thus, a developer can decide to validate objects for completeness while the model is being designed and as such, the completeness validation rules are applied to the objects. Moreover, validation for completeness may also be performed automatically before deployment using the completeness validation rules. Therefore, the Applicants respectfully request the rejection to be withdrawn.

Regarding claim 10, the Examiner states that the clause "the object instance" refers to other items in the claim. Applicants point out that claim 8, upon which claim 10 depends, includes the feature of "an instance of a meta metadata object." The clause "the object instance" refers to the instance of the meta metadata object. Therefore, the Applicants respectfully request the rejection to be withdrawn.

Regarding claim 14, the Examiner states in claim 14, completeness validation rules are applied to the object instance if a user selects validation, and completeness validation rules are also applied to the object instance automatically. The Examiner states that it is unclear how the one or more completeness validation rules is applied to the object instance. (Office

Action, p. 3). Applicants refer the Examiner to paragraph [0009] of the application for example, which states that

> In one aspect, at design time, only correctness type validation is performed. Thus, the present invention advantageously allows for incomplete objects to be created at design time. The developer, however, in this case may opt to perform completeness validation at any time. In general, a developer may opt to perform completeness and/or correctness validation at any time independent of deployment processing. In another aspect, full validation (*e.g.*, completeness and correctness) is automatically performed on the objects prior to deployment processing. (emphasis added).

Thus, a developer can decide to validate objects for completeness while the model is being designed and as such, the completeness validation rules are applied to the objects. Moreover, validation for completeness may also be performed automatically before deployment using the completeness validation rules. Therefore, the Applicants respectfully request the rejection to be withdrawn.

III. Rejection under 35 U.S.C. §102

Claims 8-14 were rejected under 35 U.S.C. §102(e) as being anticipated by Wall et al. (US Patent 7,028,288) (hereinafter "Wall"). Applicants respectfully submit that Wall does not disclose each element of the amended claims. For example, Applicants' claim 8 recites a computer-implemented method for object model design and validation, the method comprising:

> creating an instance of a meta metadata object of an object model in response to user input;
> automatically applying one or more correctness type validation rules to the object instance by confirming the object instance complies with the one or more correctness type validation rules;
> if a user selects validation of the object instance, applying one or more completeness validation rules to the object instance; and
> automatically applying both the one or more correctness validation rules and the one or more completeness validation rules to the object instance prior to deployment of the object instance at runtime. (emphasis added).

Applicants submit that several of the features recited in claim 1 are not taught by Wall. For example, claim 8 specifically recites a "method for object model design and validation," and "creating an instance of a meta metadata object of an object model in response to user input." Applicants submit that this feature recited in claim 8 is not taught by Wall.

The Examiner states that a class derived from the model class of Wall is considered as being equivalent to "a meta metadata object." Applicants respectfully disagree.

Wall describes validation of data entered into an input field of a GUI, where the data validation is performed the Controller of an MVC software application. (Wall, col. 4, lines 37-40). Wall describes

> In the IMVC software application, in accordance with one or more embodiments of the invention, data validation rules associated with an input field are embodied by a constraint (or set of constraints) applied to the Model corresponding to the input field. For example, a GUI has an input field for entering integer data (e.g., an account number entry field) having a corresponding Model, e.g., a derived class of the Model class. A data validation rule associated with the account number entry field is embodied by the constraint, and the constraint is applied to the derived class. For example, the data validation rule may state that the account number may be no longer than six digits. Therefore, the constraint (e.g., a LengthConstraint), is applied to the derived class corresponding to the account number entry field, thereby becoming an applied constraint. The applied constraint is visible to the Controller, and thus the Controller may perform data validation using an input field constraint mechanism in order to enforce the data validation rule. (emphasis added). (Wall, col. 5, lines 10-27).
>
> The input field constraint mechanism enforces the data validation rule. For example, if the data validation rule requires that the user enter only one of a certain set of states (e.g., CA, CO, NJ, NY, or PA), then the input field constraint mechanism may be implemented by the View via a drop-down list box that includes only the states CA, CO, NJ, NY, or PA as possible choices. Thus, data validation is implicit in the View, and the data validation rule is enforced. (Wall, col. 6, lines 50-57).

Applicants submit that Wall does not describe object model design and validation. Wall is directed to validation of data according to particular rules. Validating data, as described by Wall, and validating objects when creating an object model are very different. Wall makes no mention or suggestion of object model design and validation. Furthermore, Wall does not describe "creating an instance of a meta metadata object of an object model in response to user input." Wall describes that the View can create a derivative Person Model Class that correspond to input fields of the GUI. (Wall., col. 7, lines 13-18). The input fields that are presented in the GUI enforce the data validation rules. (Wall, col. 6, lines 50-57). When a user makes a selection through the GUI, there is no object that is being generated. As such, Wall fails to teach "creating an instance of a meta metadata object of an object model in response to user input."

Applicants further submit that Wall fails to teach "if a user selects validation of the object instance, applying one or more completeness validation rules to the object instance." Wall teaches that the View enforces the constraints on input fields in the GUI. (Wall, col. 9, lines 6-9). The user does not have any control over enforcing constraints. Instead, the user's actions themselves are constrained by the rules. The user in Wall does not select to have objects validated or to have constraints applied to the objects. As such, Wall fails to teach "if a user

selects validation of the object instance, applying one or more completeness validation rules to the object instance," as recited in claim 8.

Applicants submit that Wall fails to teach "automatically applying both the one or more correctness validation rules and the one or more completeness validation rules to the object instance prior to deployment of the object instance at runtime." As previously mentioned, Wall does not describe that object models can be designed or validated. The processes as described by Wall occur during runtime of the IMVC software application. (Wall, col. 10, lines 5-8). Accordingly, Wall does not teach "automatically applying both the one or more correctness validation rules and the one or more completeness validation rules to the object instance prior to deployment of the object instance at runtime."

As such, Wall cannot anticipate Applicants' claim 8, or the claims that depend therefrom. Independent claim 14 recites limitations that similarly are not disclosed by Wall, such that Wall cannot anticipate claim 14. Applicants therefore respectfully request that the rejection with respect to these claims be withdrawn.

IV. Rejection under 35 USC § 103, Wall in view of Raghuvir

Claims 1-3, 6 and 7 are rejected under 35 U.S.C. §103(a) as being unpatentable over Wall in view of Raghuvir et al. (US Publication 2004/0249823 A1) (hereinafter "Raghuvir") Claim 1 is allowable as Wall and Raghuvir either alone or in combination, do not teach or suggest each and every element of claim 1. For example, claim 1 recites in part: a system for object model design and validation, the system comprising:

> a client device configured to receive user input and provide a user interface to a user;
> a database for storing objects corresponding to an object model and for storing metadata objects describing the object model while designing the object model;
> a configuration management module for creating a deployable collection of metadata objects from the metadata objects stored in the database, wherein the deployable collection represents a tree of metadata objects starting at a root metadata object; and
> a validation engine for validating the metadata objects stored in the database by confirming the metadata objects comply with one or more validation rules, wherein said validation engine is configured to perform completeness validation on the deployable collection in response to a user entered command to perform validation on the validation subject, to automatically perform correctness validation on the deployable collection when the subject is created or updated, and to automatically perform completeness and correctness validation on the deployable collection when requested by the configuration management module. (emphasis added).

Applicants submit that Wall does not render claim 1 obvious, either alone or in combination with Raghuvir. As previously discussed, Wall does not describe object model design and validation. Wall is directed to validation of data according to particular rules. Validating data, as described by Wall, and validating objects when creating an object model are very different. Wall makes no mention or suggestion of object model design and validation. Furthermore, Wall does not describe "creating an instance of a meta metadata object of an object model in response to user input." Wall describes that the View can create a derivative Person Model Class that corresponds to input fields of the GUI. (Wall., col. 7, lines 13-18). The input fields that are presented in the GUI enforce the data validation rules. (Wall, col. 6, lines 50-57). When a user makes a selection through the GUI, there is no object that is being generated. As such, Wall fails to teach "creating an instance of a meta metadata object of an object model in response to user input." Along similar rationale, Applicants submit that Wall does not teach or suggest "a database for storing objects corresponding to an object model and for storing metadata objects describing the object model while designing the object model," as recited in claim 1. The Examiner asserts that Wall shows a database that stores Person model objects, which teaches this feature. (Office Action, p. 4). Applicants respectfully disagree. The database in Wall is not a database that is used during a design phase of an object model. Applicants direct the Examiner to paragraphs [007] and [009] of the application as filed for exemplary discussion of designing an object model. Likewise, Wall also fails to describe a validation engine as recited in claim 1. As previously discussed, Wall makes no mention or suggestion of validating metadata objects. Instead, Wall describes validation of data.

Applicants submit that the combination of Wall and Raghuvir fails to teach or suggest "a configuration management module for creating a deployable collection of metadata objects from the metadata objects stored in the database, wherein the deployable collection represents a tree of metadata objects starting at a root metadata object." The Examiner recognizes that Wall does not teach this feature. (Office Action, p. 5). Raghuvir is relied upon for such teaching.

Raghuvir does not make up for the deficiencies in Wall with respect to claim 1. Raghuvir distinguishes between a framework that provides core functionality and an application

that extends that basic functionality, and describes modeling an environment for the development of an application. (Raghuvir, [0042]. Raghuvir also describes that an object browser can enable application developers to model development objects for a business logic layer and a presentation layer of a three tier application architecture. (Raghuvir, [0053] and [0057]). Although Raghuvir describes that business objects can be modeled, there is no mention or suggestion of a deployable collection of metadata objects representing a tree of metadata objects starting at a root metadata object, as recited in claim 1. Nor is there any mention of validating the metadata objects. Thus, Raghuvir does not disclose, "a configuration management module for creating a deployable collection of metadata objects from the metadata objects stored in the database, wherein the deployable collection represents a tree of metadata objects starting at a root metadata object; and a validation engine for validating the metadata objects stored in the database," as recited in claim 1. Thus, Raghuvir cannot render claim 1 obvious either alone or in combination with Wall. As claim 15 is allowable, dependent claim 15 is also patentable for at least the same rationale. Applicants therefore respectfully request that the rejection with respect to the pending claims be withdrawn.

V. Newly Presented Claims 15 and 16

Claims 15 and 16 have been added to cover different aspects of the present invention. These claims are supported by the specification and do not add new matter. The elements recited in claims 15 and 16 are believed to be allowable over the cited art for at least a similar rationale as discussed with respect to claims 1 and 8, and others. Applicants therefore respectfully request consideration and allowance of newly presented claims 15 and 16.

VI. Amendment to the Claims

Unless otherwise specified, amendments to the claims are made for purposes of clarity, and are not intended to alter the scope of the claims or limit any equivalents thereof. The amendments are supported by the specification and do not add new matter.

CONCLUSION

In view of the foregoing, Applicants believe all claims now pending in this Application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested.

If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 650-326-2400.

Respectfully submitted,

/Naya M. Chatterjee-Marathe/

Naya M. Chatterjee-Marathe
Reg. No. 54,680

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, California 94111-3834
Tel: 650-326-2400
Fax: 415-576-0300
Attachments
NMC:pas
61413389 v1